использования ткани в трёхмерных моделях показаны на рис. 9.58. Для создания модели полощущегося флага использовалось силовое поле типа **«ветер»** (англ. *wind*).

Рис. 9.58

Мягкие тела (англ. *soft bodies*) — это специальный аппарат для реалистичного моделирования движения тел, обладающих упругостью. Без него (вручную) было бы практически невозможно грамотно построить анимацию, в которой предметы сталкиваются, сминаются, отскакивают друг от друга и т. п.

Когда для сеточной модели устанавливается свойство **«мягкое тело»**, грани приобретают свойства пружинок. Их жёсткость можно регулировать с помощью настроек на панели **Мягкое тело** (Soft Body).

Рендеринг

Конечный результат анимации — это видеоролик, записанный в одном из видеоформатов, например AVI[1], MPEG, QuickTime, Ogg Theora. Рендеринг происходит покадрово, т. е. сначала программа строит полное изображение кадра 1, затем — кадра 2 и т. д. Важная характеристика видео — частота кадров (англ. **FPS** — *frames per second*, кадры в секунду). Обычно в кино используется частота 24 кадра в секунду[2], при которой человеческий глаз воспринимает смену кадров как непрерывное движение. В этом случае для создания ролика, длящегося 10 секунд, нужно построить анимацию из 240 кадров.

[1] Строго говоря, формат AVI — это *контейнер*, внутри которого видео и звук могут быть упакованы с помощью различных кодеков (алгоритмов сжатия).

[2] В телевизионном стандарте PAL, который принят в Европе, используется частота 25 кадров с секунду, а в стандарте NTSC (США, Канада) — около 30 кадров в секунду.

Рендеринг сложных сцен, включающих миллионы граней, может занимать значительное время (дни и недели!) и требовать больших вычислительных мощностей компьютера. Прежде всего, важно быстродействие процессора и объём оперативной памяти. Поэтому рендеринг видеороликов выполняется, как правило, на нескольких мощных компьютерах, объединённых в локальную сеть. Существуют организации, предоставляющие такие услуги **(рендер-фермы)**[1].

В программе Blender режимы рендеринга настраиваются на странице свойств **Рендер**. На панели **Вывод** (Output) нужно выбрать каталог для сохранения видеоролика, формат и имя файла. На панели **Размеры** (Dimensions) задаются размеры изображения, номера начального и конечного кадров анимации, а также частота кадров. По умолчанию установлена частота 24 кадра в секунду.

Рендеринг запускается с помощью кнопки **Анимация** (Animation) на странице свойств **Рендер** или комбинацией клавиш Ctrl+F12.

Вопросы и задания

1. Расскажите об основных принципах анимации по ключевым кадрам.
2. Как можно изменять поведение объектов в промежуточных кадрах?
3. Объясните, как выполняется анимация с ключевыми формами.
4. Что такое арматура и зачем она нужна?
5. Сравните анимацию по ключевым формам и анимацию с помощью арматуры. Когда удобно использовать каждый из этих способов?
6. Объясните различие между связями «прямая кинематика» и «обратная кинематика».
7. Расскажите о возможностях моделирования физических процессов в программах трёхмерной графики.
8. Что такое система частиц и зачем она используется?
9. Как строится анимация ткани?
10. Вспомните, где ещё в компьютерной технике используется кэширование.
11. Что такое «мягкие тела»?
12. Почему рендеринг видеороликов представляет собой серьёзную проблему? Как она может быть решена?

[1] Например, http://farmerjoe.info и http://renderfarm.fi

Задачи

1. Постройте простую анимацию одного из объектов-примитивов, изменяя его положение, углы поворота и размеры.
2. Постройте анимацию улыбающегося рта.
3. Постройте фигуру шахматного короля, управляемого арматурой. Создайте небольшую анимацию, в которой король наклоняется в разные стороны.

*4. Добавьте волосы на голову обезьянки с помощью системы частиц.

*5. Постройте анимацию прыгающего теннисного мяча.

*6. Постройте анимацию флага, полощущегося на ветру.

§ 75
Язык VRML

Как вы уже знаете, трёхмерные сцены хранятся в компьютере как векторные изображения. Каждый объект в векторном формате задаётся координатами своих вершин и свойствами (например, характеристиками материала). Эти данные могут храниться во внутреннем («машинном») представлении, однако для ручной обработки (и для передачи через Интернет) удобнее, если 3D-модель представляет собой обычный текст без оформления (англ. *plain text*).

В этом параграфе мы кратко познакомимся с **языком VRML** (Virtual Reality Modeling Language — язык моделирования виртуальной реальности), который позволяет сохранить трёхмерную сцену в текстовом файле и потом просматривать её в специальной программе или в веб-браузере (с помощью дополнительного модуля — *плагина*). Язык VRML «понимают» многие программы трёхмерного моделирования, в том числе системы автоматизированного проектирования (САПР). С помощью VRML сделаны виртуальная экскурсия по мемориалу на Мамаевом кургане и 3D-модели исторических событий (например, полета в космос Юрия Гагарина) на сайте фирмы Parallel Graphics[1].

С помощью VRML можно описать не только форму трёхмерных объектов, но и их физические свойства: цвет, текстуру, блеск, прозрачность. Объекты могут быть гиперссылками на другие документы. Язык VRML позволяет создавать анимацию, менять освещение, включать и выключать звуки, а также добавлять к сцене программный код на языках Java или JavaScript. Современный стандарт для работы с трёхмерными моделями — X3D считается наследником языка VRML и использует хранение данных в XML-формате.

[1] http://www.parallelgraphics.com/products/showroom/event/

Трёхмерную сцену в VRML называют **миром** (англ. *world*), поэтому VRML-файлы имеют расширение wrl. Это обычные текстовые файлы, которые можно редактировать в любом текстовом редакторе. Кроме того, существуют специальные VRML-редакторы, например WhiteDune[1].

Для просмотра VRML-моделей нужно установить соответствующий плагин к браузеру (например, Cortona3D Viewer[2]) или самостоятельное приложение (например, кроссплатформенную программу view3dscene[3]).

Построим с помощью VRML трёхмерную модель комнаты. Её стены (а также пол и потолок) можно представить в виде плит (блоков, параллелепипедов). Пусть длина каждой стены — 4 м, высота — 3 м, а толщина — 0,1 м. Тогда стена в плоскости XOY (рис. 9.59) описывается так:

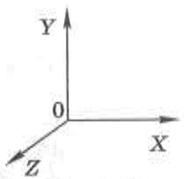

Рис. 9.59

```
#VRML V2.0 utf8
Shape
  {
  geometry Box { size 4 3 0.1 }
  }
```

В первой строке указана версия языка VRML (2.0) и кодировка текста (utf-8 — это один из вариантов UNICODE). У единственного безымянного объекта `Shape` (англ. *shape* — форма, фигура) задано только одно свойство — `geometry` (геометрия). Слово `Box` означает, что эта фигура — параллелепипед («коробка»), его размеры (по осям *X*, *Y* и *Z* соответственно) указаны после слова `size` (размер).

Объекты **сцены** в VRML называются **узлами**. Названия классов объектов начинаются с заглавной буквы, а названия свойств **(полей)** — со строчных. В нашем примере объект класса `Shape` имеет поле `geometry`, определяющее форму тела. Значение этого

[1] http://vrml.cip.ica.uni-stuttgart.de/dune/

[2] http://www.cortona3d.com/cortona

[3] http://castle-engine.sourceforge.net/view3dscene.php

поля — объект класса `Box` (параллелепипед)[1]. Узел `Box`, в свою очередь, имеет поле `size`, определяющее размеры плиты по каждой координатной оси.

Кроме параллелепипедов в языке VRML есть объекты других классов, например `Sphere` (шар), `Cylinder` (цилиндр), `Cone` (конус). Более сложные фигуры строятся из отдельных граней.

Объекты VRML имеют довольно много полей. Если значение поля не указано, используется некоторое стандартное значение (значение по умолчанию). Например, по умолчанию тело располагается так, чтобы его центр совпадал с началом координат, и равномерно «покрашено» в белый цвет.

Приведённый выше VRML-код можно записать в файл (назвав его, например, `first.wrl`) и загрузить в программу-просмотрщик. При этом мы не просто увидим объёмную стену, но и сможем «походить» вокруг неё. В программе view3dscene это будет выглядеть так, как показано на рис. 9.60.

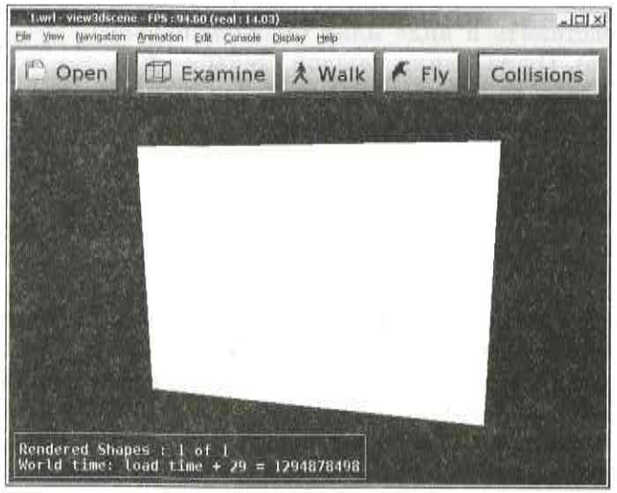

Рис. 9.60

Программа должна выполнять рендеринг для загруженной трёхмерной модели очень быстро, поэтому качество картинки будет существенно хуже, чем в профессиональных программах 3D-моделирования типа Blender или 3ds Max.

[1] Каждому объекту можно присвоить собственное имя, но делать это не обязательно.

В нашем файле точка наблюдения не задана, и программа самостоятельно выбирает её (по умолчанию). Положение этой точки можно задать явно, добавив в VRML-файл код:

```
Viewpoint
  {
  position 0 0 5
  }
```

Здесь свойство `position` (положение) объекта `Viewpoint` (точка наблюдения) определяет начальные координаты наблюдателя. В данном случае он находится в точке с координатами $X = 0$, $Y = 0$ и $Z = 5$.

Существует несколько режимов просмотра трёхмерной сцены:

- **Examine** (рассматривать, исследовать) — наблюдатель неподвижен, объект можно поворачивать и рассматривать под разными углами;
- **Walk** (ходить, делать обход) — сцена неподвижна, а наблюдатель перемещается внутри неё «по поверхности земли»;
- **Fly** (летать) — отличается от предыдущего режима «выключенной» силой тяжести.

Кнопка **Collisions** (столкновения) определяет, может ли наблюдатель проходить сквозь стены и предметы (если она отключена, то может).

Чтобы придать стене более реальный вид, надо заполнить у объекта `Shape` еще одно поле — `appearance` («внешность», «наружность»), отвечающее за то, как объект выглядит. Все свойства, влияющие на внешний вид объекта, собраны в единый объект с именем `Appearance`. Зададим для стены материал (свойство `material` объекта `Appearance`) и установим для этого материала цвет (свойство `diffuseColor`):

```
#VRML V2.0 utf8
Shape
  {
  geometry Box { size 4 3 0.1 }
  appearance Appearance
    {
    material Material { diffuseColor 0.7 0.7 0.7 }
    }
  }
```

Числа после названия свойства `diffuseColor` задают цвет стены в виде трёх составляющих модели RGB — красной, зелёной и синей, каждая из которых принимает значение от 0 до 1 (это соответствует диапазону от 0 до 255 при целочисленном RGB-кодировании цвета). В данном случае все они равны 0,7 — это светло-серый цвет.

Вместо того чтобы «красить» стену, можно было «оклеить» её обоями, т. е. наложить рисунок (текстуру). Для этого нужно задать свойство `texture` (текстура):

```
#VRML V2.0 utf8
Shape
  {
  geometry Box { size 4 3 0.1 }
  appearance Appearance
    {
    texture ImageTexture { url["texture.png"] }
    }
  }
```

В данном случае текстура относится к классу `ImageTexture` (текстура-рисунок) и находится в файле `texture.png` в текущем каталоге. Рисунок растягивается на всю поверхность, поэтому его пропорции должны соответствовать соотношению размеров стены.

Остальные стены строятся аналогично, но с одним существенным отличием: их придётся смещать в пространстве (иначе по умолчанию их центр будет расположен в начале координат). Предположим, что созданная стена будет «дальней от нас». Тогда «левая» стена имеет размеры 0,1 × 3 × 4 м (по осям X, Y и Z соответственно), и её нужно передвинуть на 2,05 м влево вдоль оси X и на 1,95 м «в сторону наблюдателя» вдоль оси Z (величина 0,05 м — это половина толщины стены!).

Для перемещения объектов в виртуальном пространстве используется узел `Transform` (превращать, изменять). Он способен выполнять для присоединённых к нему узлов (перечисленных в списке `children` — «потомки», «подчинённые объекты») следующие действия (и любые их комбинации!):

- `translation` (смещение) — смещение центра объекта вдоль каждой из трёх осей координат;
- `rotation` (вращение, поворот) — поворот объекта вокруг трёх осей координат;
- `scale` (изменение масштаба) — умножение размеров объекта на коэффициенты, отдельно по каждой оси.

В нашем случае требуется только перемещение. В список `children` (он записывается в квадратных скобках) мы включим одну фигуру (объект `Shape`) — параллелепипед, изображающий левую стену. Этот код нужно добавить в конец VRML-файла:

```
Transform
  {
  translation -2.05 0 1.95
  children
    [
    Shape
      {
      geometry Box { size 0.1 3 4 }
      }
    ]
  }
```

Теперь вы можете самостоятельно построить «виртуальную комнату» целиком.

Вопросы и задания

1. Как вы понимаете термин «виртуальная реальность»?
2. Какие свойства трёхмерных объектов можно моделировать с помощью VRML?
3. Что представляет собой VRML-файл? Какое расширение он имеет и почему?
4. Какое программное обеспечение требуется для работы с VRML?
5. Сравните системы координат, которые используются в математике и в VRML 2.0.
6. Объясните, что такое сцена, узлы и поля. Приведите примеры.
7. Что такое значение по умолчанию? Какие преимущества даёт такой способ задания значений при передаче VRML-модели по сети?
8. Каково по умолчанию положение тел относительно начала координат? Что нужно сделать, чтобы изменить положение объекта?
9. Перечислите режимы просмотра трёхмерных сцен. Чем они различаются?
10. Как называется узел VRML, отвечающий за внешний вид объектов?
11. Каким образом кодируется цвет в языке VRML?
12. Как нанести текстуру на поверхность объекта?
13. Какие действия можно выполнять с помощью узла `Transform`?
14. Обсудите, где можно использовать язык VRML.

Задачи

1. Сделайте комнату замкнутой: опишите все четыре стены, пол и потолок. Поскольку по умолчанию точка наблюдения будет вне комнаты, подумайте, как попасть внутрь (используйте кнопку **Collisions**).
2. Подготовьте и наложите на каждую стену отдельную текстуру.
3. Напишите программу, которая создаёт VRML-файл по введённым размерам стен комнаты.

*4. Напишите программу, которая создаёт VRML-файл с описанием шахматной доски, состоящей из 64 чередующихся чёрных и белых блоков (объектов `Box`).

5. Постройте простейший лабиринт из нескольких коридоров. Пройдите его от точки входа до точки выхода. Используя режим полёта (**Fly**), посмотрите на лабиринт сверху.
6. Используя комбинацию простейших геометрических тел, попробуйте создать какие-нибудь простые объёмные предметы. Например, конус и пара цилиндров позволяют «построить» ракету, а из сфер разного радиуса можно создать модель планетной системы.
7. Используя блоки (параллелепипеды), постройте объёмные буквы «Г», «Е» и «Ш».

*8. Найдите информацию о полях узла `Material` и посмотрите, как их значения влияют на изображение объекта.

*9. Найдите информацию об узле `Transform`. Примените режимы `rotation` и `scale`.

*10. Напишите VRML-код, который строит снеговика.

Практические работы к главе 9

Работа № 77 «Управление сценой»
Работа № 78 «Работа с объектами»
Работа № 79 «Сеточные модели»
Работа № 80 «Модификаторы»
Работа № 81 «Пластина»
Работа № 82 «Тела вращения»
Работа № 83 «Материалы»
Работа № 84 «Текстуры»
Работа № 85 «UV-развёртка»
Работа № 86 «Рендеринг»
Работа № 87 «Анимация»
Работа № 88 «Анимация. Ключевые формы»
Работа № 89 «Анимация. Арматура»
Работа № 90 «Язык VRML»

ЭОР к главе 9 на сайте ФЦИОР (http://fcior.edu.ru)

- Изображения. Виды
- Общие понятия об аксонометрических проекциях
- Проекции простейших геометрических фигур на плоскость

Самое важное в главе 9

- Трёхмерные сцены строятся с помощью векторной графики.
- Трёхмерные модели объектов состоят из отдельных граней (полигонов), чаще всего треугольных или четырёхугольных. Каждый полигон ограничен рёбрами.
- Для каждой грани можно независимо задавать свойства материала — цвет, блики, шероховатость и т. п. Можно наносить на грани рисунок — текстуру.
- Рендеринг — это построение плоского изображения трёхмерной сцены с учётом материалов, текстур, освещённости, свойств внешней среды. При этом выполняются сложные математические расчёты хода большого количества лучей от источников света, поэтому для рендеринга сложных сцен нужен быстродействующий процессор и большой объём оперативной памяти.
- Анимация трёхмерных сцен строится по кадрам: человек вручную определяет положение объектов в ключевых кадрах, а все промежуточные кадры строятся автоматически.
- Для хранения трёхмерных сцен можно использовать как двоичные, так и текстовые файлы.